Contextual Saliency for Detecting Anomalies within Unmanned Aerial Vehicle (UAV) Video

Simon Gökstorp
Department of Computer Science
Durham University
Durham, United Kingdom
simon.gokstorp@durham.ac.uk

***Abstract*—Unmanned Aerial Vehicles (UAV) can be used to great effect for the purposes of surveillance or search and rescue operations. UAV enable search and rescue teams to cover large areas more efficiently and in less time. However, using UAV for this purpose involves the creation of large amounts of data (typically video) which must be analysed before any potential findings can be uncovered and actions taken. This is a slow and expensive process which can result in significant delays to the response time after a target is seen by the UAV. To solve this problem we propose a deep model using a visual saliency approach to automatically analyse and detect anomalies in UAV video. Our Contextual Saliency for Anomaly Detection in UAV Video (CoSADUV) model is based on the state-of-the-art in visual saliency detection using deep convolutional neural networks and considers local and scene context, with novel additions in utilizing temporal information through a convolutional LSTM layer and modifications to the base model. Our model achieves promising results with the addition of the temporal implementation producing significantly improved results compared to the state-of-the-art in saliency detection. However, due to limitations in the dataset used the model fails to generalize well to other data, failing to beat the state-of-the-art in anomaly detection in UAV footage. The approach taken shows promise with the modifications made yielding significant performance improvements and is worthy of future investigation. The lack of a publicly available dataset for anomaly detection in UAV video poses a significant roadblock to any deep learning approach to this task, however despite this our paper shows that leveraging temporal information for this task, which the state-of-the-art does not currently do, can lead to improved performance.**

***Index Terms*—computer vision, image processing, machine learning, deep learning, image saliency, aerial vision, UAV image processing, UAV object detection and tracking, saliency detection, contextual saliency**

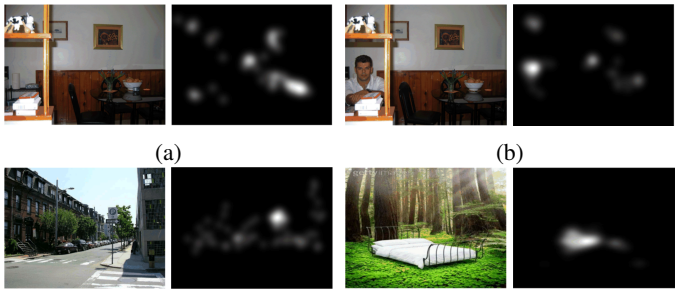
I. Introduction

Search and rescue operations can be very costly and time-consuming due to the large and often difficult to access areas involved. For example, the search and rescue operation following the disappearance of Steve Fossett in September 2007 involved more than twenty aircraft scouting an area over 52000 square kilometres, as well as over 50000 volunteers analysing 2 million satellite photos [1]. The operation cost the state of Nevada nearly $\$700000$, yet Steve Fossett was not found until a year later when a hiker stumbled upon some items from his airplane [2].

Modern advances in technology have enabled the use of Unmanned Aerial Vehicles (UAV) for the purposes of surveillance and search and rescue operations, reducing the costs and improving the capabilities of such operations. UAV can cover large distances and areas quickly and efficiently; however, processing and analysing the video recorded by UAV is still a costly and time-consuming task. Time is often critical to the outcome of search and rescue operations, so an automated solution which reduces the cost and increases the speed of this analysis would be beneficial for this task.

Visual saliency is a measure of the conspicuity of objects in an image, meaning how much they stand out from the image or how unique they are [3]. Through the application of visual saliency detection, computer vision systems are capable of identifying and extracting the most distinctive parts of an image. However, while this can relatively easily be applied to the tasks of identifying people or vehicles in an image, applying it to the task of anomaly detection in UAV video is more complicated. The general scope of detecting 'anomalies' means that the proposed solution should be capable of detecting a wide range of objects, not just people. Additionally, the type of saliency expected in UAV footage differs from traditional examples considered in the field of visual saliency detection [4]. These approaches often consider high-quality images with a single or few salient objects, which are unobstructed and often located near the centre of the image. On the other hand, salient objects captured in UAV footage can be located anywhere in the image, may be obstructed by other objects such as foliage, and the footage itself can be grainy. This means that the salient objects we wish to detect are often less conspicuous than in the typical case.

Contextual saliency is an extension of visual saliency which considers the context of an image in determining the salient objects in it. Two cases showing the impact of context on saliency detection are presented in Figure 1. First, Figure 1a and 1b show the effect of global context on saliency detection in an image. The two stimulus images are the same except for the man present in the left part of the second stimulus image. This change drastically alters the saliency levels across the image, showcasing that it is not enough to consider the local context (i.e. a small area around a given pixel or object) to determine visual saliency. The second case, in Figure 1c and 1d, shows the effect of scene context on visual saliency. In the left image pair, the most salient object is the traffic sign



(a) (b) (c) (d)

Fig. 1: The effect of context on saliency detection in images, from [5]. Each pair of images shows the stimulus image on the left and the corresponding saliency map on the right. (a) and (b) show the effect of global context, while (c) and (d) show the effect of scene context.

near the centre of the image, while in the right image pair the bed is the most salient object as it appears out-of-place in a forest. The context of the scene the image depicts, in this case a street in the left image pair and a forest in the right image pair, also affects what objects are the most salient. It is therefore important to consider both these types of context when evaluating visual saliency.

There are also many challenges associated with handling UAV footage which do not need to be considered in the more general case of salient object detection. These include potentially large scale variations due to altitude changes and noise in the image due to poor camera quality, motion of the UAV, image compression, and more. All of these aspects complicate the task of surveillance or search and rescue to find survivors, wreckage, or any other anomaly.

Previous approaches to the problem of anomaly detection in UAV video have relied on classical computer vision techniques to achieve saliency detection, for example colour space manipulation in [4] and image region segmentation in [6]. Those approaches achieving the best results are generally very slow, taking several minutes to process each frame, and they often do not scale well with larger image sizes [7]. This makes them unsuitable for practical applications in search and rescue operations, as the target objects are often very small and captured from a high altitude, meaning the solution must be capable of handling large image sizes. Additionally, if the solution takes a long time to process each image it will not be a suitable replacement for manual analysis as the results would be produced too slowly. Many classical approaches also commonly disregard the temporal information in video by processing each frame independently, meaning they are not optimised for processing video.

In order to solve this problem, in this paper we propose a novel Contextual Saliency for Anomaly Detection in UAV Video (CoSADUV) model based on the DSCLRCN model of [5], part of the state-of-the-art in salient object detection using a deep convolutional neural network (CNN) approach. CNN are commonly used in computer vision tasks to process images. Each convolutional layer works by applying a set of convolutions with learnable weights across the image, extracting features from it. Convolutional layers can be stacked to extract more complex features, yielding 'deep' CNN, and can be applied to images without requiring massive amounts of neurons and memory which a network of fully connected layers would require since the same convolution kernels are applied across the image.

While the state-of-the-art in salient object detection has recently been dominated by such deep learning approaches, no such approach has previously been applied to the task of salient object detection in UAV images or video. Our proposed model considers both local context in the image as well as the general scene context of the image as a whole. It is novel in leveraging the temporal information carried in video captured by UAV, which previous approaches have not done, by using a convolutional LSTM (convLSTM) layer to propagate saliency feature information through time. In this way we present a novel approach to the problem of anomaly detection in UAV video by using contextual saliency and temporal information. Through our novel changes to the model architecture our proposed CoSADUV model achieves significantly improved performance over the DSCLRCN model for this task, but does not generalize well to other data and therefore fails to beat the state-of-the-art.

II. Related Work

A vast amount of work has been carried out in investigating the use of saliency for anomaly detection in UAV video, with numerous different approaches and areas being considered. The seminal and related previous work is grouped into three main groups and presented below. Please note that while in some works a technical distinction is made between the terms 'UAV' and 'drone' they are considered to be interchangeable for this work.

A. Saliency Detection

One of the first and most seminal works on visual saliency detection is [3], which has served as the basis and inspiration of many more recent methods. In this work, Itti et al. proposed a bottom-up approach to saliency detection, i.e. an approach based on low-level features such as intensity, colour and orientation. This approach was inspired by neuroscience principles and combined the aforementioned features at multiple scales into a single saliency map, finally using a neural network to select the most salient areas. More recently, [8] built on this traditional approach by including object-level information to refine the result of the bottom-up approach by removing noise salient regions. This is accomplished by using a bottom-up approach to create a saliency map, then performing object proposal on that map and finally combining these to produce the output saliency map with most noise salient regions removed.

Seeking to further improve the bottom-up approach of [3], [9] observed that all bottom-up methods rely on some assumptions about the higher-level information, such as the *"contrast and compactness of the salient objects and the*

background." These assumptions lead to common limitations in such methods, for example selecting only the most distinctive parts of salient objects rather than the entire object for local-contrast based methods, failing to accurately detect multiple salient objects for global-contrast based methods and struggling to accurately select salient objects that touch the edge of the image for methods based on a boundary-prior (i.e. an assumption made about the boundary of the salient object).

In order to overcome these limitations [9] proposed a method that utilizes a recursive processing step in order to enhance the output saliency map and reduce dependency on the initial saliency map produced by the bottom-up approach. This method achieved better results than state-of-the-art methods regarding detecting multiple salient objects and the precision of the object areas, even when applied to UAV/aerial-style images. However, this approach still suffered the same shortcomings regarding reliance on priors, detecting objects that touch the edges of the image and detecting smaller and more subtle objects, although to a lesser extent. Additionally, the approach suffered from over-detection in UAV/aerial-style images, leading to a large number of false positives being produced and the authors made a comment that all tested methods suffered a noticeable reduction in performance when applied to these images compared to non-UAV images.

B. Contextual Saliency

Another source of information in images that can be applied to visual saliency detection is the context of the image. This is information about the general contents of the image as a whole, for example the terrain, environment or conditions displayed in the image, or the presence of additional objects in other areas of the image. One of the earliest usages of context for automated saliency detection is [10]. They observe that most previous methods had only considered appearance information, such as colour or texture, and ignored higher-level information like context. Some methods utilized context in the form of user input or assumptions made about positioning of the salient objects, and thus were not fully automated. Their model utilized an "auto-context classifier" to learn the context of a salient object through iterative learning, allowing that object to be efficiently detected in future images [10]. However, due to this design efficient salient object detection of a given object was only possible after carrying out the process of iterative learning for that object.

Another application of contextual saliency was suggested by [11], which utilized contextual salient information for mobile image retrieval. Their method extracts contextual information of "multi-photos" (multiple photos of the same object) in order to select salient features [11]. Since all photos in the multi-photo contain the salient object the feature points corresponding to this object will occur more commonly than others and some will occur in all or nearly all of the photos. These are detected by performing SIFT point matching for each image-image pair and used to extract the salient object. The work of [12] integrated contextual information with the traditional bottom-up method of Itti et al., using the context to dynamically weight the low-level features used in the bottom-up approach. This use of context had at the time not been considered in previous works and resulted in a model better suited for task-driven scenarios such as target detection due to the inclusion of higher-level information.

More recently, [5] sought to use contextual information together with a neural network based approach for saliency detection, proposing a *"deep spatial contextual long-term recurrent convolutional network (DSCLRCN)."* This model consists of a CNN used to extract local image features, which is trained to automatically extract features such as colour, intensity, faces, etc. from raw image data. Next, a long short-term memory (LSTM) neural network is trained to remember *"long-term spatial interactions,"* and thus learn the global context of the image [5]. This model evaluates saliency per pixel in the image, while still considering the global context, achieving better results than all previous models on eye-fixation datasets.

A similar approach was presented by [13] which also uses a CNN for contextual saliency detection. Compared to other state-of-the-art deep neural network methods at the time, this approach was simpler, while still considering local and global information in the image. The suggested approach achieved results on par with the state-of-the-art in a fraction of the time (speed-up by *"a factor of 18 to 100"*), allowing near real-time saliency detection [13]. This was accomplished through using a *"single and fully convolutional CNN,"* and avoiding any special/expensive pre- or post-processing [13]. However, this approach was not designed for nor tested on UAV footage, and resizes images to 352×352 for evaluation, potentially losing out on small-scale information and context which could be very important for UAV images. It also did not consider the scene context of image unlike [5], instead processing only local and global context within the image.

C. Saliency for UAV Video

Images and video captured by UAV commonly feature a set of distinct properties when compared to the images considered in most saliency detection approaches. As mentioned previously, these include potentially being grainy, low-quality and noisy (from the motion of the UAV, encoding/transmission of the image, etc.), the possibility of being captured at varying altitudes (and thus scales) and speeds (and thus level of motion noise) and more. Additionally, the salient objects in typical images considered for saliency detection are often large in the image and placed at or near the centre However, the salient objects in UAV images are typically very small and scattered across the image. Figure 2 shows an example UAV image considered in this area. There are several salient objects in this image, in this case a group of people. An ideal saliency detector for such images would detect all these objects, despite their small scale and them being scattered around the image.

These properties often place limitations on the application of saliency detection methods to these images, but may also be exploitable by a model specifically designed for this type of imagery. One such method is proposed by [4], which is

Fig. 2: An example UAV image considered for the task of anomaly detection, from HERIDAL dataset [14].

partially based on the bottom-up approach of [3]. This method uses a combination of low-level contrast features, mean-shift segmentation with additional histogram information and multichannel edge features [4]. The method achieves good results, but is tailored for *"rural, uncluttered and relatively uniform environments,"* and does not consider any contextual information [4]. This was built upon by [6], who commented that the previous method was very slow and had a high false positive rate. The proposed solution is targeted towards detecting people and vehicles on roads, utilizing region segmentation and extraction to improve the produced saliency map. This method achieved good results for the targeted scenario, but struggled with images where the salient objects were not clearly situated in road regions. Additionally, while the scenario involves UAV video, no temporal information is leveraged in the proposed method.

An alternative approach was taken by [7], who analysed the current state-of-the-art methods for saliency detection when applied to UAV images. They considered some methods specifically designed for UAV images, such as [4], but these were disregarded for various reasons such as time performance and scalability for larger images. Out of the four surveyed methods, the model using low-level features based on wavelet transform [15] achieved the best results (90% Recall, 44% Precision for 39 target objects over 10 complex images), although with a poor execution time of 148s per image [7]. With parallelisation the method was optimised to 5.38s per image. This analysis was, however, conducted over a very small sample size (10 simple images and 10 images featuring complex scenes) and none of the considered methods utilized contextual or temporal information, so there is likely room for an advanced solution tailored for UAV images to achieve better results.

Very recently the work of [7] was built upon by [14]. Based on their findings, [14] present an approach that uses the wavelet transform-based model in [15] to produce a saliency map which is used to select the 300 most salient patches in the image. Next, a CNN trained to detect people is applied to each patch. In order to train and test this model, they produce a new dataset for UAV for search and rescue purposes. Their model achieves state-of-the-art results on this new dataset, achieving a higher precision but lower recall score than a Faster R-CNN model, presented by [16], trained on the same dataset. However, the approach presented in this work ignores contextual information in the image, as the saliency model it uses does not consider contextual saliency. The model is also only designed for the scope of detecting people in land-based situations and is therefore not directly generalizable to the more general task of anomaly detection. It additionally elects to not utilize temporal information, instead only considering individual images and frames.

Considering temporal information could massively benefit any saliency approach that is designed for video. A model for video saliency prediction for non-UAV videos is presented by [17], which utilizes a deep CNN. This model extracts "deep features" from the video and combines these with spatial-temporal object candidates [17]. A random forest classifier is trained to assign saliency to each object candidate. When tested on a typical saliency detection dataset, this method achieved better results than the current state-of-the-art. However, this approach was designed for the typical saliency detection case, not for UAV video, and it is likely to exhibit the same shortcomings as other typical saliency approaches when applied to UAV video. Another approach was taken by [18], who used an architecture called convolutional LSTM, created by [19], to process spatiotemporal information in video. Their model uses on bidirectional processing, requiring the entire video to be input for optimal results. As with many other video saliency models this model does not consider the context of the video, and is designed for relatively small images sizes (473×473). No previous approaches were found which were designed to utilize this temporal information in UAV video.

Previous methods for saliency detection in UAV images and video are generally limited in scope, not considering contextual or temporal information available, or making assumptions about the type of salient object or environment expected. While there has been a large amount of research into the topics of contextual saliency and video saliency, these ideas have not been extensively applied to UAV video. In the field of visual saliency detection deep learning models are dominating the state-of-the-art, both in terms of accuracy and execution speeds. Generally, approaches utilizing contextual saliency by considering the local and/or global context in images achieve higher performance than those that do not.

Our proposed CoSADUV model is novel in applying these idea to the topic of anomaly detection in UAV video. It does so by building on the DSCLRCN model of [5]. This work is chosen as it is at the state-of-the-art in the field of contextual saliency detection and we judged that augmenting a contextual saliency approach with temporal processing is easier and more likely to succeed than vice-versa. Additionally, this work has source code publicly available, which we believe significantly speeds up development of the proposed solution, and the chosen model was designed to process larger images than many other state-of-the-art approaches such as [13], which

we judge to be especially important for UAV video. It also includes scene context, which we consider to be important for achieving a generalizable solution to this problem. By adapting this model by replacing the last convolutional layer with a convolutional LSTM layer and changing the activation function of the last convolutional layer as well as the loss function we produce our novel CoSADUV model, which achieves significantly improved salient object detection performance in UAV video compared to the base DSCLRCN model.

III. Solution

Our proposed solution is a deep CNN model based on the state-of-the-art in contextual saliency detection. The model is adapted to the task of anomaly detection in UAV video by changing the activation function as well as the loss function used in when training the model. It additionally utilizes temporal information carried in video by propagating data through time to improve the analysis of subsequent frames via a convolutional LSTM layer. More detail about the proposed solution is provided below.

A. Deep Learning Model

Based on the results of the literature survey, we chose to construct the solution based on the state-of-the-art deep learning model for contextual saliency proposed by [5]. This choice was made because the survey of related works revealed that deep learning models generally outperform classical computer vision approaches, both in terms of accuracy and execution speeds. As the Deep Spatial Contextual Long-term Recurrent Convolutional Network (DSCLRCN) model presented in the work of [5] was implemented using MATLAB and the Caffe deep learning framework and our programming language of choice for this task was Python, we developed the code for our model based on an unofficial re-implementation[1] of the DSCLRCN model in Python using the PyTorch deep learning framework. The structure of the DSCLRCN model is shown in Figure 3. The model contains three main sections, outlined below.

The first section applies a series of convolutions in order to extract local features across the image. This is achieved by adapting the 50 layer version of the 'residual network' approach for image recognition proposed by [20], or alternatively the 16 layer VGG model for image recognition proposed by [21]. Through experimentation [5] found that using the residual network model produced better results for the DSCLRCN model. Therefore only this approach is outlined below and considered in this work.

The work of [20] noted that deeper networks were more difficult to train than shallower ones, and surprisingly as networks were made deeper their performance plateaued before rapidly worsening. In order to solve this problem they proposed adding residual blocks, which feature 'shortcut connections' that skip layers in each block to produce the output of each block as the output of the last layer within the block plus the input. Using such blocks simplifies the learning process of the network, making deeper networks easier to train. The work produced several models of varying depths, with the deeper models providing better results, except for at extreme depths.

The DSCLRCN model uses one of these models, ResNet-50, with pretrained weights to extract image features. Using this model provides a balance of high performance in image identification and execution speed. The model is adapted by using the first 49 convolutional layers of the model and removing the last fully connected layer, thus utilizing the feature extraction framework of the model for a different task than image recognition. It is reasoned that the features the model has learned to extracted for the task of image recognition will be similar to those that are important for saliency detection. By using a state-of-the-art model for feature extraction with pretrained weights, the training of the entire model can be sped up, as only fine-tuning of the feature extraction weights is needed.

The 49 convolutional layers extracted from the ResNet-50 model are structured in five convolution blocks, all except the first containing between three and six residual blocks. The first convolution block has a stride of 4, the second a stride of 1, the third a stride of 2, and the fourth and fifth are adapted to have strides of 1, resulting in an overall stride of 8 across the 49 convolutional layers. The final convolution block produces a feature map with 2048 channels, which is reduced to 512 through a single 1×1 convolution layer with rectified linear (ReLU) activation function to reduce the complexity and make the features easier to learn by subsequent layers. Finally, an L_2-normalisation layer is applied to normalise the scale of the feature map. The ResNet-50 model is pretrained on the Imagenet dataset presented in [22] for image recognition. More detail of the architecture of the (unmodified) ResNet-50 model can be found in [20]. Further information regarding the modifications as well as figures summarising the final structure of the modified ResNet-50 model are included in [5].

The modified feature extractor used has an combined stride of 8 in both the horizontal and vertical direction, and produces a feature vector of length 512 at each spatial location. Thus, if the input image is of size $W \times H \times 3$, the output of the local feature extractor will be of size $\frac{W}{8} \times \frac{H}{8} \times 512$. The DSCLRCN model uses input colour images of size 640×480, producing a feature map of size $80 \times 60 \times 512$, but the architecture can be generalized to any size.

The second section in the DSCLRCN model uses a scene recognition model to extract the scene context of the image. The model used is a pretrained CNN model presented in [23]. This is a model based on the architecture of [24], which is trained on the Places database for scene recognition of [25]. From this model the convolutional layers are used to extract scene features in the image, and then a fully connected layer with 128 neurons and ReLU activation function is applied to produce a scene context vector. The scale of this vector is then also normalised through an L_2-normalisation layer. This ensures that the scene context vector can subsequently be processed with the feature map. More information about

[1]Source code for Python re-implementation of the DSCLRCN model used is available at: https://github.com/AAshqar/DSCLRCN-PyTorch

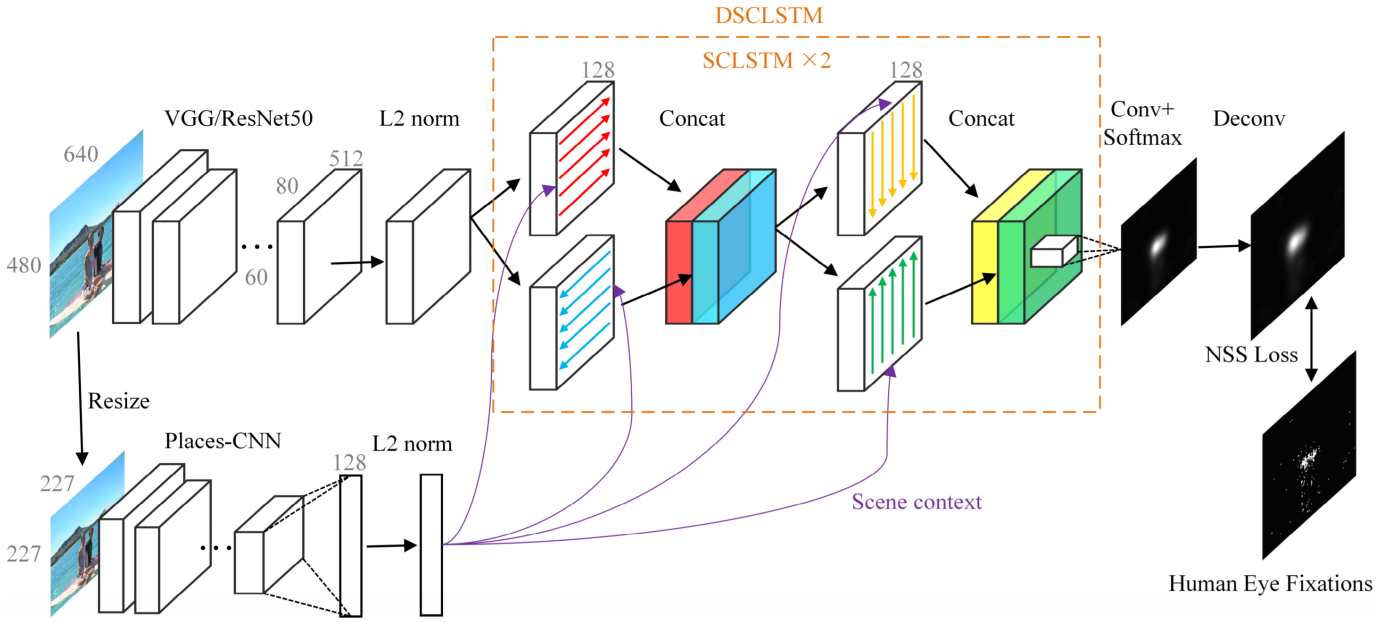


Fig. 3: An overview of the architecture of the DSCLRCN model of [5]. Note that both VGG-16 and ResNet-50 were investigated for local feature extraction by [5], but only ResNet-50 is considered in this work as this produced better saliency results.

the Places database can be found in [25], and the full details of the Places-CNN scene recognition model used can be found in [23].

The scene context is extracted and included in the subsequent saliency computation as another representation of the context of the image. As shown in Figure 1 not only the local and global context of the image have significant effects on visual saliency, but also the context of the scene of the image. By considering this information, the saliency detection model is better equipped to handle the varying scenes and locations that saliency models are usually tested on. This is also relevant for the use of the model with UAV images, where what is considered salient in an image is hugely reliant on the scene and terrain in the image. For example, while a deep blue jacket spotted in a forest would be considered extremely salient, the same colour spotted while surveying the ocean likely would not be. By extracting the local context and the scene context using two separate models, both of which are pretrained for their task, both models are focused on a singular task, and are subsequently fine-tuned for these tasks. This approach is thus likely to produce better results than if a single model were used to extract local features which were also used to evaluate the scene context.

The third section in the DSCLRCN model is the so-called Spatial Contextual Long Short-Term Memory (SCLSTM) block. The basis of this block is the Long Short-Term Memory (LSTM) network. This is a recurrent neural network architecture, capable of propagating information forward along some dimension, for example in time. The LSTM architecture was suggested by [26] as a method of solving the vanishing gradient problem that traditional RNN suffer from. Each neuron in an LSTM layer has a memory cell c, which stores information between time steps, a hidden state h, which is given as the output at each time step, and four gates which control how the memory cell is updated and the hidden state calculated. The four gates are the input gate i, the forget gate f, the output gate o, and the input modulation gate g. The computation of the memory cell and the hidden state through these gates for each neuron at a time step t are as follows:

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o) \tag{3}$$

$$g_t = \phi(U_g x_t + W_g h_{t-1} + b_g) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t g_t \tag{5}$$

$$h_t = o_t \phi(c_t) \tag{6}$$

where b denotes a bias term, U denotes the input weights, W denotes the recurrent weights, σ denotes a sigmoid function, and ϕ denotes a hyperbolic tangent (tanh) function. The equations for σ and ϕ are provided below in Equation 7 and Equation 8, respectively.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

$$\phi(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{8}$$

Through Equations 1 to 6 the input modulation gate g controls how to update the memory cell c, as it has a range of $(-1, 1)$. The input gate i controls how much the memory cell is updated by g. The forget gate f controls how much information is remembered by the cell state in each time step,

and the new cell state is computed by a combination of these three gates and the previous cell state c_{t-1}. Lastly, the output gate o controls how much information is output form the cell state into the hidden state h at each time step, which is then provided as the output of the neuron. More information about LSTM networks and these equations can be found in [27].

An extension of the LSTM network is the bidirectional LSTM (BLSTM). This is realised as two LSTM applied to the same sequence in opposite directions. For each time step t, the output of the BLSTM is the output of the first LSTM concatenated with the output of the second LSTM. Thus, if each LSTM has N neurons in the final layer, the output of the BLSTM would have $2N$ channels. The BLSTM construct allows information to be propagated along the sequence in both directions.

In the DSCLRCN model, LSTM networks are applied to the local features extracted by the ResNet-50 model in the first step, yielding a Spatial LSTM (SLSTM). Specifically, first the image is split into rows, with each pixel in the row being considered a time step t. Next, a BLSTM is applied to the feature map to scan it horizontally, both from left-to-right and right-to-left, propagating feature information along both directions. The output from the BLSTM is then subsequently split into columns. A second BLSTM is applied to each column, with each pixel in the column being considered a time step, propagating feature information vertically, both top-to-bottom and bottom-to-top. By the end of this process, all feature information has been propagated from each pixel in the image to every other pixel, thus encompassing the global context in the image at each spatial location. The SLSTM takes as input the feature map produced by the local feature extractor in the first step of the DSCLRCN model. Given an input feature map of dimensions $\frac{W}{8} \times \frac{H}{8} \times 512$, the SLSTM produces an output of dimensions $\frac{W}{8} \times \frac{H}{8} \times 2N$, where N is the number of neurons in the last BLSTM. For this work, all BLSTM are built with 128 neurons. Higher neuron counts yielded only minor improvements in accuracy, while causing a significant increase in the complexity of the model, and thus its memory requirement and run time for training and evaluation.

The SLSTM construct is modified to also include the scene context extracted by the Places-CNN model, yielding a Spatial Contextual LSTM (SCLSTM). In [5], this is done by modifying the LSTM gate equations for the first input to each LSTM to include the scene context vector as an additional term, replacing Equations 1 to 4 with Equations 9 to 12:

$$i_1 = \sigma(U_i x_1 + W_i h_0 + V_i s + b_i) \tag{9}$$

$$f_1 = \sigma(U_f x_1 + W_f h_0 + V_f s + b_f) \tag{10}$$

$$o_1 = \sigma(U_o x_1 + W_o h_0 + V_o s + b_o) \tag{11}$$

$$g_1 = \phi(U_g x_1 + W_g h_0 + V_g s + b_g) \tag{12}$$

where V represents the additional scene context weights, and s represent the scene context vector produced by the scene context identifier. Note that here $t = 1$ denotes the first input of the sequence, and h_0 is the initial hidden state of the LSTM. Through these equations the scene context is included in the first time step of each sequence, that is the first pixel of each row when the feature map is processed horizontally and the first pixel of each column when the feature map is processed vertically, in both directions. The scene context is then propagated to the other pixels in the sequence through the memory of the LSTM, thus being included at each spatial location in the feature map.

For this work, however, the scene context is included in a slightly different way. Due to limitations in the implementation of the DSCLRCN model used, we were unable to modify the LSTM gate equations as outlined above. Instead, two fully-connected layers were added, each taking the 128 length scene context vector as input. The first fully-connected layer has 512 neurons, and is applied to the scene context before the first BLSTM is applied to the feature map. The fully-connected layer is used to change the shape of the scene context vector to match the shape of the local feature vector at each spatial location. The 512-length scene context vector is then added to the local feature vector of the first pixel in each row when the left-to-right LSTM is applied, and to the last pixel in each row when the right-to-left LSTM is applied. The second fully-connected layer has 256 neurons, and is applied to the scene context vector before the second BLSTM is applied. A separate fully-connected layer is necessary as after the first BLSTM each pixel has a feature vector of length 256. The same procedure used for the first BLSTM is repeated for the other BLSTM, incorporating the scene context vector in each application of the BLSTM. The output of the second fully-connected layer is reused for each subsequent BLSTM, as all BLSTM are constructed with 128 neurons, and thus produce output with a channel dimension of size 256. The gate equations used for the first pixel in each sequence are thus:

$$i_1 = \sigma(U_i \hat{x}_1 + W_i h_0 + b_i) \tag{13}$$

$$f_1 = \sigma(U_f \hat{x}_1 + W_f h_0 + b_f) \tag{14}$$

$$o_1 = \sigma(U_o \hat{x}_1 + W_o h_0 + b_o) \tag{15}$$

$$g_1 = \phi(U_g \hat{x}_1 + W_g h_0 + b_g) \tag{16}$$

$$\hat{x}_1 = x_1 \oplus Vs \tag{17}$$

where x_1 and s are defined as before, $\hat{x}_1$ is the modified feature vector of the first pixel after the addition of the scene context vector, V is the scene context weights, applied by the fully-connected layer, and $\oplus$ denotes element-wise addition. By modifying the LSTM gate equations for the first pixel of each sequence in this way, the scene context weights V_i, V_f, V_o and V_g used in the original DSCLRCN model are each approximated as:

$$V_{[i,f,o,g]} s \approx U_{[i,f,o,g]} Vs \tag{18}$$

for each gate i, f, o and g. While this only approximates the method used by [5], this was the best solution found that

avoids massively increasing the implementation difficulty of our model.

The Spatial Contextual Long Short-Term Memory unit forms the backbone of the DSCLRCN model, achieving a contextual saliency by propagating the global and scene context to all spatial locations in the image. In order to allow the model to learn more complex relationships in the local features and scene context extracted from the image, two SCLSTM units are applied in sequence. By stacking the SCLSTM units, a Deep Spatial Contextual Long Short-Term Memory (DSCLSTM) unit is formed. Adding further SCLSTM units to increase the depth of the model was found by [5] not to increase performance further, while increasing complexity and training time of the model. The second SCLSTM unit takes as input the output of the last BLSTM in the previous SCLSTM unit, and includes the scene context in the same way. By using BLSTM units with 128 neurons in the second SCLSTM also, the depth of the feature map produced is not increased by the second SCLSTM. The shape of the final output of the DSCLSTM unit is therefore $\frac{W}{8} \times \frac{}{8} \times 256$.

After the DSCLSTM is applied the output is reduced to a single channel by applying a convolutional layer with a kernel size of $(1, 1)$. This convolution kernel evaluates the saliency at each spatial location in the feature map output based on the processed feature information produced by the DSCLSTM. After this a *Softmax()* activation function is used. This is an activation function commonly used for multi-class classification tasks. It takes as input a list of values in the range $[-\infty, \infty]$, where each value corresponds to a class, and outputs a list of probabilities corresponding to each class. As the *Softmax()* function outputs probabilities, each value will be in the range $[0, 1]$, and all the values in the list will sum to 1. More detail about the *Softmax()* activation function can be found in [27].

The *Softmax()* activation function is used to produce competition amongst the pixels in the image. As the outputs of the *Softmax()* function will sum to 1, any increase in saliency of one pixel will cause a decrease in the saliency level of all other pixels. In this application each pixel is can be though of as a different class, with the output of the *Softmax()* function at each pixel corresponding to the probability of that pixel being the most salient pixel in the image. The *Softmax()* function also normalises the output values into the range $[0, 1]$.

In the final step of the DSCLRCN model a deconvolution with stride 8 is applied in order to increase the size of the image from $\frac{W}{8} \times \frac{H}{8}$ to $W \times H$. This upsampling is done through bilinear interpolation, and is not learnable by the model, as suggested by [28]. This produces an output saliency map of the same spatial shape as the input image, with a single channel with the saliency prediction. The saliency prediction map is then ready to be compared with the corresponding ground truth image.

B. Modifications for UAV Data

In order to adapt the DSCLRCN model for use with UAV images some modifications were made to the model architecture and training procedure. Firstly, the activation function applied to the output of the final convolution layer was changed from the *Softmax()* function to the *Sigmoid()* function. Although the lateral competition introduced by the Softmax*Softmax()* function is desirable as it helps produce cleaner saliency predictions, it has the side-effect that the magnitude of the output is always the same. A model using the *Softmax()* activation function as the last activation function is therefore unable to output an output that contains no predicted saliency for an input image. The model is also incapable of predicting the overall saliency level of an image (i.e. whether the image contains many salient objects or very few salient objects, the magnitude of the saliency prediction will remain the same).

This is not an issue for the case of typical visual saliency prediction, as the model should predict the most salient item in every image. Such cases therefore have no negative examples (i.e. images with no salient objects in them). This is however an issue for applying saliency prediction for anomaly detection, as the model should be able to predict a lack of any salient objects in an image such as the one in Figure 4. This image contains no objects that are salient for our task of anomaly detection, despite the presence of objects that might normally be considered salient. In order to solve this problem we replace the *Softmax()* activation function after the last convolution layer with a *Sigmoid()* activation function, given previously in Equation 7. As the *Sigmoid()* function has a range of $(0, 1)$, it is well suited for tasks that evaluate probabilities. By applying this activation function to the output of the last convolutional layer, each pixel in the output is assigned a value in this range, corresponding to the saliency of that pixel. As the *Sigmoid()* function is applied to each pixel individually, no constraints are placed on the image as a whole, or on the relationship between pixels. The model is thus able to output a low value at every pixel in the image if it does not detect any salient objects.

Fig. 4: An example image captured by UAV with no salient object (for the purpose of anomaly detection). Image taken from UAV123 dataset [29].

We also adjusted the testing procedure used when validating and testing the model. The authors of [5] found that applying a Gaussian blur to the saliency prediction produced by the DSCLRCN model improved its performance by smoothing out

the saliency response. Such blurring may improve the saliency prediction for large objects by removing large peaks and small gaps in the prediction but it also removes the detail at the smaller scales in the image. In UAV video target salient objects can be present in varying scales due to factors such as the altitude of the UAV. We therefore omit this stage of processing in order to preserve small-scale detail in the predictions.

In addition to the changes made to the architecture and post-processing of the model we also changed the loss function used to train the model. To train the DSCLRCN model [5] used the negative Normalized Scanpath Saliency (NSS) proposed by [30] to compute the loss of a prediction with respect to the ground truth from human eye fixation data. Given a saliency prediction x and the corresponding ground truth y NSS is calculated as:

$$NSS(x, y) = \frac{\sum(\bar{x} \circ y)}{\sum y} \tag{19}$$

$$\bar{x} = \frac{x - \mu(x)}{\sigma(x)} \tag{20}$$

where $\circ$ denotes the element-wise product, $\bar{x}$ is the saliency map of x normalised to have a mean of 0 and a standard deviation of 1, $\mu(x)$ denotes the mean of x, and $\sigma(x)$ denotes the standard deviation of x.

The NSS score measures how much the mean predicted saliency value of target pixels in x (i.e. pixels where the corresponding pixel in y have a value above 0) are higher than the mean of the saliency prediction x as a whole. By using the normalised version of x, $\bar{x}$, the NSS score is measured in units of standard deviations. This score has a range of $(-\infty, \infty)$ with a positive score indicating a correlation between the target pixels and higher than average saliency predictions, a score of 0 indicating no correlation, and a negative score indicating an anti-correlation. For example, an NSS score of 2 means that, on average, the predicted saliency values at target locations are 2 standard deviations above the mean saliency value of the prediction. As training of the network is done through loss minimisation, the negative NSS score is used as the loss function.

Using the negative NSS score as the loss function for training and testing of the model produces a similar problem to the use of the *Softmax()* activation function discussed above. Namely, the NSS function requires the presence of target pixels in the ground truth y. If there are no targets in the ground truth fixation data, as could be the case in the data considered for UAV anomaly detection, then $\sum y$ is 0, and thus the NSS is not defined. Therefore, we are unable to use the negative NSS loss function for training our model, while including images that contain no salient object in the dataset.

Another loss function that was considered by [5] for training of their model is the Pearson's Correlation Coefficient (CC). This function, along with NSS, was recommended for use for saliency prediction evaluation by [31]. This function considers the saliency prediction x and the ground truth saliency map y to be random variables, and provides a measure in the range of $[-1, 1]$ of how strongly they correlate:

$$CC(x, y) = \frac{cov(x, y)}{\sigma(x) \times \sigma(y)} \tag{21}$$

where $cov(x, y)$ denotes the covariance of x and y, and σ denotes the standard deviation as above. A value of 1 or -1 denotes perfect positive or negative correlation, respectively, while a value of 0 denotes no correlation. The loss is therefore returned as the absolute value of $CC(x, y)$. This function, however, suffers from the same problem as the NSS score. If y has no saliency targets, there will be no variance in y, and it will have a standard deviation of 0. Like the NSS, the CC is therefore undefined for images where the ground truth y has no salient objects.

In order to solve this problem, we investigated several other loss functions for training our model for anomaly detection in UAV video. First, based on the recommendation of [18] we used a compound loss function of the Cross Entropy (CE) as well as the Mean Absolute Error (MAE), also known as L_1-loss of the predicted saliency x compared to the ground truth y. Given x and y that both contain values in the range $[0, 1]$, the CE of x and y is computed as:

$$CE(x, y) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(x_i) + (1 - x_i) \log(1 - y_i)] \tag{22}$$

where N is the total number of pixels in the images, e.g. 640×480, and x_i and y_i represent corresponding pixels in x and y. The Cross Entropy loss function has a range of $[0, \infty)$, with a lower value corresponding to a larger similarity between x and y. The CE function produces larger loss values as x_i and y_i differ more, with the loss rapidly increasing as the difference approaches 1. It is therefore very sensitive to outliers. The MAE of x and y is computed as:

$$MAE(x, y) = \frac{1}{N} \sum_{i=1}^{N} |y_i - x_i| \tag{23}$$

where N, x_i and y_i are as above. The Mean Absolute Error loss grows linearly with the difference between x_i and y_i, and as such is less susceptible to outliers. MAE has a range of $[0, 1]$, with 0 meaning there is no difference between x and y. Finally, the loss is computed as the sum of the CE of x and y and the MAE of x and y:

$$CE\_MAE(x, y) = CE(x, y) + MAE(x, y) \tag{24}$$

By combining these two loss functions in this way, [18] found that their model for video salient object detection achieved better results as the compound loss function better captured different factors contributing to the overall quality of the results.

A second loss function that was investigated was a modified version of Normalized Scanpath Saliency. We noted that the NSS loss function had been used to great success in recent

works, and is recommended for evaluating saliency predictions by many surveys of common metrics such as the work of [31] and [32], which found that out of nine scores surveyed NSS performed the most consistently with human evaluations. For these reasons, we wished to apply the NSS loss function to our task of anomaly detection in UAV video, while still being able to include negative images in the dataset. Our chosen approach for this was to use the NSS loss function when possible, and apply a different loss function when the NSS is not defined. The resultant loss function is computed as:

$$NSS_{alt}(x,y) = \begin{cases} -\frac{\sum(\bar{x} \circ y)}{\sum y} & \sum y > 0 \\ \sigma(x) & \sum y = 0 \end{cases} \tag{25}$$

where $\bar{x}$ is calculated as per Equation 20, and σ denotes the standard deviation. Note that unlike in Equation 19 the negative sum is returned if $\sum y > 0$, and so NSS_{alt} is used directly as the loss function. The rationale behind the design of this function is that if there is no target salient object in the ground truth y, then the model should output a predicted saliency map that is monotonous and invariable across the image, as there are no spatial locations in the image that are more salient than the others. Although this loss function is likely imperfect, and is not well balanced between the two cases as the ranges of them are significantly different, this alteration of NSS allows us to apply the NSS loss function to our UAV data.

The last loss function that was investigated was also inspired by the Normalized Scanpath Saliency function, although it is significantly different. We took the idea of NSS to measure the mean predicted saliency value at target salient points, but rather than normalising the saliency prediction to a mean of 0 and a standard deviation of 1, we introduce a second term in the form of the mean predicted saliency value at non-target points. This loss function, which we named Difference of Means (DoM), is computed as:

$$DoM(x,y) = \mu(x_i, y_i = 0) - \mu(x_i, y_i > 0) \tag{26}$$

where $\mu(x_i, y_i = 0)$ denotes the mean value of the set of pixels in x where the corresponding location in y has a value of 0. The creation and investigation of this loss function was inspired by the observance that the dataset used in training our model for anomaly detection in UAV video contained a large number of frames with a very small, singular target. The targets in these frames were significantly smaller than targets in typically considered images. This meant that when trained with the CE_MAE loss function described above the model was able to achieve a very low error by outputting low saliency predictions throughout the image. This observation is further discussed in Section VI. This issue led us to want a loss function where the task of predicting high saliency at the salient object locations and the task of predicting low saliency at non-salient locations were balanced, rather than each pixel being treated as equal. Additionally, this loss function has an advantage in that it is applied equally to all images and ground truths, unlike the NSS_{alt} loss function which uses a piecewise function to handle ground truths with no salient locations.

In order to speed up the learning process we used the Adam optimiser proposed by [33]. This is a variation of the Stochastic Gradient Descent (SGD) optimization method, which was used by [5] to train the DSCLRCN model. The Adam optimiser is better suited for tasks with large data and/or parameters, and is widely used in training deep models for computer vision tasks. When training with Adam we used a learning rate of 0.01, a β_1 of 0.9 and a β_2 of 0.999. We also used a learning rate scheduler to reduce the learning rate by a factor of 2.5 every epoch, which was also done by [5]. This helps speed up the learning process by allowing us to use a high learning rate at the beginning of training, allowing the model to quickly approach a desired local minimum, and a lower learning rate later in the training process to fine-tune the model. Some models achieved a higher validation accuracy when trained with SGD. In these cases we used a learning rate of 0.01 as with Adam, a momentum value of 0.9 and a weight decay of 0.0005, as these were the training settings used in the work of [5] to train the DSCLRCN model on the SALICON dataset. For both optimisers, since pretrained weights are used for the local feature extractor and the scene context extractor models we reduced the learning rates for these layers by a factor of 0.1 compared to the rest of the model, allowing the weights to be fine-tuned for our task and reducing the risk of decay in performance of these parts of the model. This was also done in the original DSCLRCN model. Implementing the above modifications produces the non-temporal version of our proposed CoSADUV model, CoSADUV_NT

C. Temporal Implementation

As the DSCLRCN model is designed for the task of visual saliency prediction in images, it is not adapted to processing videos. We therefore further augmented the model to leverage the temporal consistency of the saliency in consecutive frames of a video, producing our proposed CoSADUV model. We achieved this by replacing the final convolution that reduces the channel dimension to 1 for saliency prediction with a convolutional LSTM (convLSTM) layer. This architecture is proposed by [19] to be used in place of traditional LSTM layers for processing images.

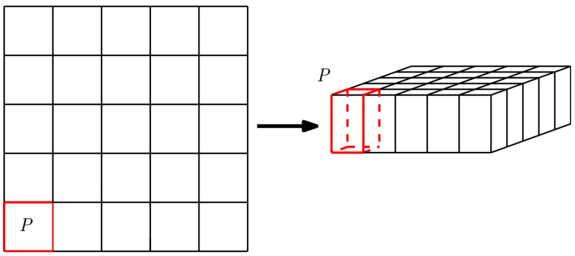


Fig. 5: Transforming the 2D input image into a 3D tensor for processing via convLSTM. Graphic from [19].

Traditional LSTM take one-dimensional input for each batch and time step, meaning they are poorly suited for

processing image data as they do not utilize spatial correlation. The convLSTM architecture instead takes as input a 3D tensor, adding two spatial dimensions. Likewise, all gates i, f, o and g, all cell states c and all hidden states h are also represented as 3D tensors rather than one-dimensional vectors. A helpful visual for the convLSTM architecture is given in [19], which is shown in Figure 5. We can imagine the inputs, cell states and hidden states as vectors P in a two-dimensional grid, with one vector at each spatial point in the image. Then, the convLSTM computes the future cell state and hidden state at each spatial point as a convolution of the inputs and past states of its local neighbourhood. The equations for computing the gates, cell state and hidden state of a convLSTM neuron are:

$$i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i) \tag{27}$$

$$f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f) \tag{28}$$

$$o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o) \tag{29}$$

$$g_t = \phi(U_g * x_t + W_g * h_{t-1} + b_g) \tag{30}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t \tag{31}$$

$$h_t = o_t \circ \phi(c_t) \tag{32}$$

where $*$ denotes convolution, and $\circ$ denotes the element-wise product as before. The spatial locality in the image is therefore applied and maintained, as the gates and states at each spatial location is computed by convolution, rather than as a function of all previous inputs and hidden states. By replacing the last convolution with a convLSTM layer, the saliency prediction at each spatial location is computed as a function of the feature vector computed by the DSCLSTM at that location and neighbouring locations, as well the previous cell and hidden states at that location and neighbouring locations. If a kernel of size 1 is used in the convLSTM, only the vector at the spatial location is considered.

We apply a convLSTM layer with 256 input channels, as produced by the last BLSTM unit, and a single output channel. As the output is produced using the *tanh()* activation function, which has a range of $(-1, 1)$, the output values are not directly suitable to be output as saliency prediction values. Since the *tanh()* function is a rescaled *Sigmoid()* function (see Equation 8 and Equation 7), we produce the saliency prediction p by mapping the output of the convLSTM layer h_t to the range $(0, 1)$ as $p_t = \frac{h_t+1}{2}$.

D. Datasets

One limitation of using a deep learning-based approach in our proposed solution is the need for large amounts of data to train, validate and test the model on. This is especially true for our task of UAV anomaly detection by saliency prediction, as there is currently no publicly available dataset designed for this task. A number of public datasets were considered for training and testing our model.

The first dataset we considered was the Saliency in Context (SALICON) dataset [34]. This dataset was used to train and test the DSCLRCN model in [5], and is a large-scale image dataset for contextual saliency tasks. This dataset was considered early on in the design process despite containing neither videos nor UAV style footage. We reasoned that by initially training the model on such a dataset and subsequently fine-tuning it on a more appropriate dataset the model would first learn to extract and process features that are important for determining saliency, and then refine this learning for the context of UAV footage. However, this idea was quickly abandoned due to the differences between typical saliency images considered in datasets such as SALICON and the UAV footage considered for our task.

The second dataset considered was the UAV123 dataset presented by [29]. This dataset is formed of 91 sequences, split into a total of 123 subsequences. The sequences are all formed of aerial footage and most are filmed from a drone, with some of the sequences being produced by simulation. The dataset is designed for object tracking tasks, and labels for this purpose for all 91 sequences are also provided. Each subsequence is between 100 and 3100 frames, with a median length of $\sim$900 frames at 30 Frames per Second (FPS). The dataset contains videos for a range of target objects such as people, cars, bikes, UAV, and buildings. In total the dataset contains over $100'000$ labelled frames. A frame from a sequence in this dataset is shown in Figure 4.

Even though the UAV123 dataset is not designed nor labelled for saliency prediction tasks, it is very useful for our task. In a significant number of the sequences in UAV123 the target object for tracking is the only salient object in the image, and therefore the provided object tracking ground truth labels closely resemble what one would expect for saliency prediction labels. This meant that we were able to use a large portion of the UAV123 dataset with our model without needing to produce our own labels for the data, saving a large amount of time. Additionally, the sequences in the UAV123 dataset include many attributes that differentiate the UAV images considered in this task from typically considered images for saliency prediction tasks, including scale variation, occlusion, negative images (i.e. no salient objects present) and clutter. These attributes should be conducive to training a model for the general task of anomaly detection in UAV video without limiting the model to a small set of objects or scenes. Lastly, there are very few datasets of UAV video publicly available, and so any data that can be used without creating our own labels is desirable.

We used the UAV123 dataset to train, validate and test our proposed model. In order to improve the quality of the dataset for the task of anomaly detection by saliency prediction, we removed all 'building' sequences as the target object, as these were deemed contrary to our goal. We also removed all 'UAV' sequences and the single 'bird' sequence due to the extreme levels of noise present in these sequences, as well as the presence of a large number of salient objects in the images that were not included in the labels. While we could have produced our own labels for these sequences and thus included all sequences in the dataset, the dataset contains enough sequences that we judged this to not be necessary. We

Fig. 6: A frame from the 'Person3_s' simulated video sequence in the UAV123 dataset by [29], which was excluded when training and testing our model.

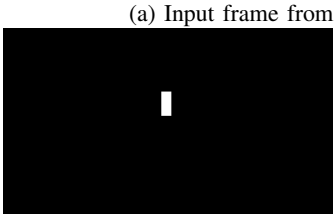
similarly removed all sequences produced by simulation, an example of which is shown in Figure 6. As can be seen in this image, although the simulations are created with modern life-like graphics, there are noticeable differences in the texture and colour palette when compared to real UAV footage, which might have a negative impact on the performance of the model.

After removing these sequences we are left with 70 sequences out of the initial 91. We split the sequences into training, validation and testing sets, with 35 sequences in the training set, 17 sequences in the validation set and 18 sequences testing sets. We assigned sequences with the same class of target object as evenly as possible, for example including the 'bike1' sequence in the training set, the 'bike2' sequence in the validation set, and the 'bike3' sequence in the testing set. Due to the large total number of frames in the dataset we cut each sequence longer than 300 frames down to the first 300 frames for training and validation, resulting in $\sim$10000 total frames in the training set and $\sim$5000 total frames in the validation set. This was done to reduce the training time of the model without further reducing the number of different sequences considered. Specifically, these values were chosen to produce a similar dataset size to the SALICON dataset used by [5] to train the DSCLRCN model.

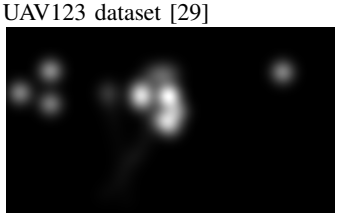
In addition to the object tracking labels provided with the UAV123 dataset by its original creators, [35] produced visual saliency labels for a subset of sequences in a new dataset named EyeTrackUAV. This data was collected using human eye fixation data collected in an experiment, similarly to how the SALICON dataset was generated. While this data is extremely useful for our task, saliency labels are only provided for 19 of the 91 sequences in the UAV123 dataset with one of the sequences being a 'building' sequence, which was disregarded for reasons outlined previously. Although the EyeTrackUAV dataset contains $\sim$26500 frames in total, the low number of sequences included means the data will be less varied than that in the original UAV123 dataset. This could lead to overfitting of the sequences in the dataset, producing a model which does not generalize to other scenarios. Additionally, as the EyeTrackUAV labels are produced from human eye fixation data, they are at times actually worse for our task than the object tracking labels provided with UAV123.

A comparison of the UAV123 and the EyeTrackUAV labels for an image in the UAV123 dataset is provided in Figure 7. This example illustrates the issues with using the EyeTrackUAV dataset for training our model. Since the data was compiled using human eye fixation data, it is generally less precise and therefore worse for the task of salient object detection. In the frame shown in Figure 7 this is clearly shown as the label from the UAV123 dataset highlights only the salient object, the person near the centre of the image, while the labels from the EyeTrackUAV dataset is less well defined. The most salient area in this label is the person; however, the area near the person is also labelled as highly salient, and there are some areas of saliency where there are no salient objects such as the grey patch of grass at the left side of the image.

(a) Input frame from UAV123 dataset [29]



(b) Object tracking label [29]



(c) Human eye fixation label [35]

Fig. 7: A comparison of the object tracking labels provided with the UAV123 dataset and the human eye fixation labels produced by [35]. The input frame is from the 'person3' sequence.

For these reasons we elected not to use the EyeTrackUAV dataset as the main dataset for training our model, instead choosing to investigate the potential of refining the model through transfer learning on the EyeTrackUAV dataset after initially training it on the UAV123 dataset. This approach is intended to overcome the limitations of the UAV123 dataset and the EyeTrackUAV dataset outlined above by using both in different stages of training. Initially training the model on the UAV123 dataset allows it to first learn local feature and scene context extraction for UAV images, as well as the relationship between global context (as extracted by the DSCLSTM units) and visual saliency. As the UAV123 dataset is labelled for object tracking, which for the selected sequences is similar to visual saliency prediction, the model learns to identify the tracked/salient object in each frame of each sequence. The

Model Architecture	Training Loss Function	↑NSS_{alt}(+)	↓NSS_{alt}(-)	↓CE	↓MAE	↓DoM
DSCLRCN	NSS	3.552	0.091	0.164	0.122	−0.398
DSCLRCN	NSS_{alt}	2.607	0.017	0.065	0.013	−0.076
CoSADUV_NT	DoM	3.315	0.163	0.181	0.039	−0.571
CoSADUV_NT †	DoM	2.699	0.209	1.672	0.249	**−0.631**
CoSADUV_NT	NSS_{alt}	1.070	0.020	0.625	0.464	−0.024
CoSADUV_NT †	NSS_{alt}	1.622	0.014	0.625	0.464	−0.036
CoSADUV_NT	CE_MAE	1.535	**0.001**	**0.050**	**0.012**	−0.002
CoSADUV (1, 1)	NSS_{alt}	0.763	0.109	0.256	0.216	−0.089
CoSADUV (3, 1)	NSS_{alt}	8.851	0.023	0.091	0.053	−0.251
CoSADUV (3, 2)	NSS_{alt}	**9.044**	0.054	3.227	0.188	−0.189

TABLE I: Performance of different models and training settings on the UAV123 test set. The NSS_{alt} metric is split into two values: the NSS score for targets containing some salient pixels, and the standard deviation for targets containing no salient pixels, indicated by (+) and (-), respectively. For each metric the arrow indicates whether a higher or lower score is better, and the best score for each metric is displayed in bold.

model then learns to generalize this for identifying salient objects when it is trained on the EyeTrackUAV dataset.

IV. Results

In order to investigate the performance of our proposed model and compare it to the base DSCLRCN model we collected results on several different models and training settings. We compare three distinctive model architectures: DSCLRCN, the baseline, CoSADUV_NT, our proposed solution without the temporal implementation, and CoSADUV, our full proposed model. For each architecture several versions were tested with variations in the loss function and hyperparameters used for training, certain network settings used and the dataset the models were trained on. We report the results of each model using several loss functions as performance metrics, all of which were discussed above. All models were tested using a GeForce RTX 2080 Ti GPU and use ∼3200MB of GPU memory and ∼3700MB system RAM with some variation between the different architectures. The proposed models achieve a processing speed of 2.2 FPS without any parallel processing. The addition of the temporal implementation does not cause a noticeable decrease in the processing speed.

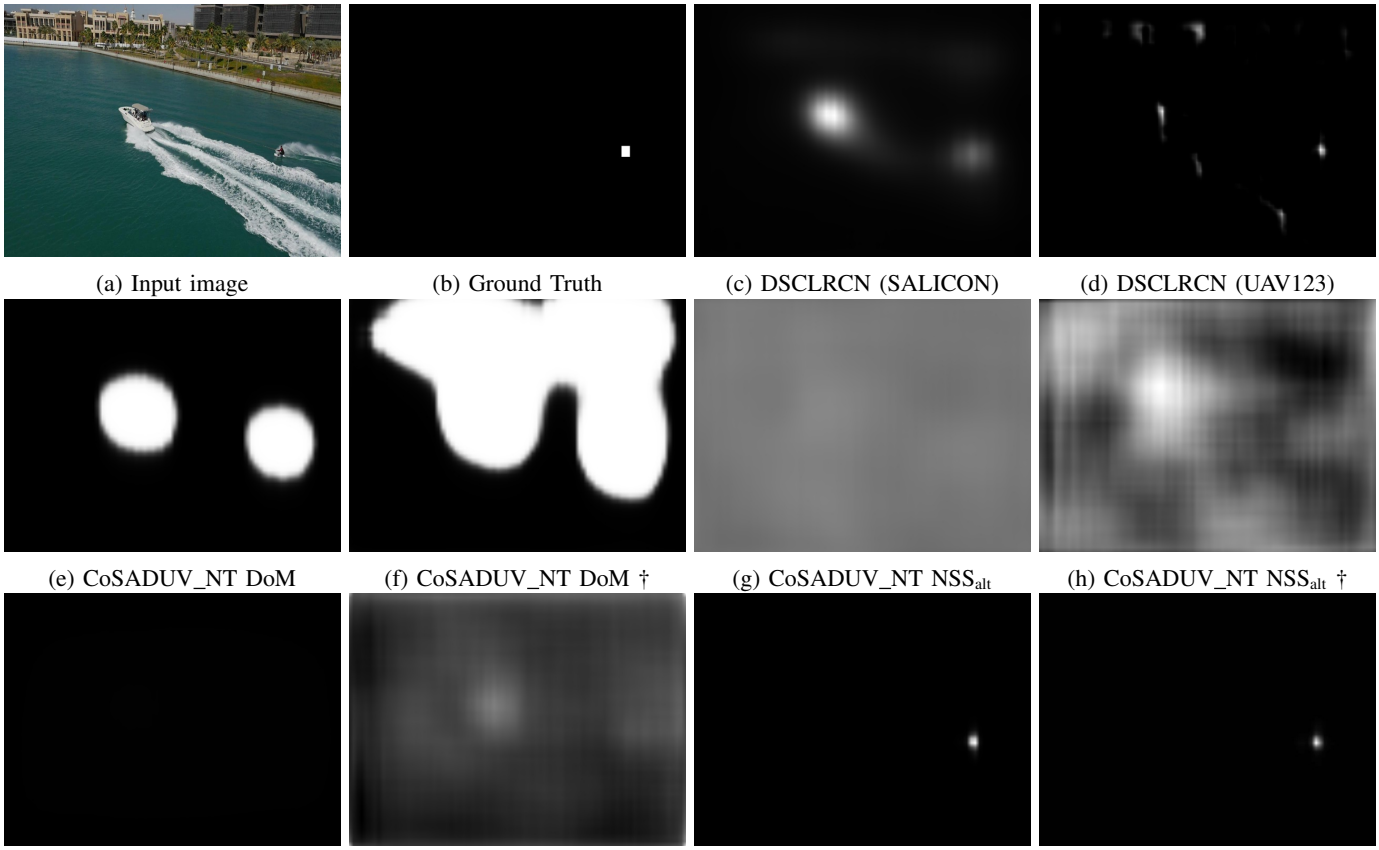
An overview of the performance of the different models on the test set we created of UAV123 sequences is shown in Table I. As outlined above, this set is formed of 18 sequences from the dataset, selected evenly from the different types of sequences. The predictions produced by the DSCLRCN models were evaluated without applying any Gaussian blurring to provide a more fair comparison between the models, and the prediction maps produced by the DSCLRCN models and the temporal CoSADUV models were scaled to the expected range of $[0, 1]$. All results were collected by resizing the input and ground truths from the provided size (1280×720) to (640×480) before applying each model.

All models are trained and validated on the UAV123 dataset using the training, validation and testing split previously discussed except for the first DSCLRCN model reported, which is trained and validated on the SALICON dataset. The DSCLRCN models and the CoSADUV_NT model trained with DoM loss function were trained using the SGD optimiser while Adam was used for the rest. Each model was trained for 10 epochs, with the best version of the model being selected via validation experiments after each epoch. Additionally, models marked with † were initially trained on the UAV123 dataset and then further trained on the EyeTrackUAV dataset via transfer learning. DSCLRCN is the model proposed by [5], used as a baseline, CoSADUV_NT is our proposed model without the temporal implementation, and CoSADUV (k, b) is our full proposed model, where k indicates the kernel size used in the convLSTM layer and b indicates the number of previous frames that the backpropagation recursed through during training.

Table I clearly shows that our proposed model achieves improved performance when compared to the baseline DSCLRCN model. The non-temporal model CoSADUV_NT trained with DoM loss function performs better than the DSCLRCN model with respect to some performance metrics, and worse with respect to others. The temporal model CoSADUV (3, 1), however, achieves significantly better performance than both of these models with respect to nearly all metrics. This quantitative result is confirmed by qualitative results analysed below.

Two of the CoSADUV models without temporal implementation were further trained through transfer learning on the EyeTrackUAV dataset. These models are labelled in Table I with the †symbol. This transfer learning resulted in a small decrease in performance for the model trained with the DoM loss function but an increase in performance for the model trained with the NSS_{alt} loss function. Notably, the CoSADUV_NT model trained with the DoM loss function and further trained via transfer learning achieves the highest score on the DoM metric.

Figure 8 provides qualitative data for the performance of the tested models on the UAV123 dataset. The input image is frame 49 of the 'wakeboard8' sequence in the dataset, which we allocated to our validation set. Although there are two salient objects in this frame (the boat near the centre and the wakeboarder near the bottom-right), only the wakeboarder is labelled in the ground truths provided with the dataset. Despite this the two DSCLRCN models successfully detect both of these object, although the model trained on the UAV123 dataset contains several areas of false positives



(a) Input image (b) Ground Truth (c) DSCLRCN (SALICON) (d) DSCLRCN (UAV123)

(e) CoSADUV_NT DoM (f) CoSADUV_NT DoM † (g) CoSADUV_NT NSS_{alt} (h) CoSADUV_NT NSS_{alt} †

(i) CoSADUV_NT CE_MAE (j) CoSADUV 1×1 (1 frame) (k) CoSADUV 3×3 (1 frame) (l) CoSADUV 3×3 (2 frame)

Fig. 8: Qualitative results of the different models tested on an image our UAV123 validation set. The input image is frame 49 in the 'wakeboard8' sequence in UAV123 [29].

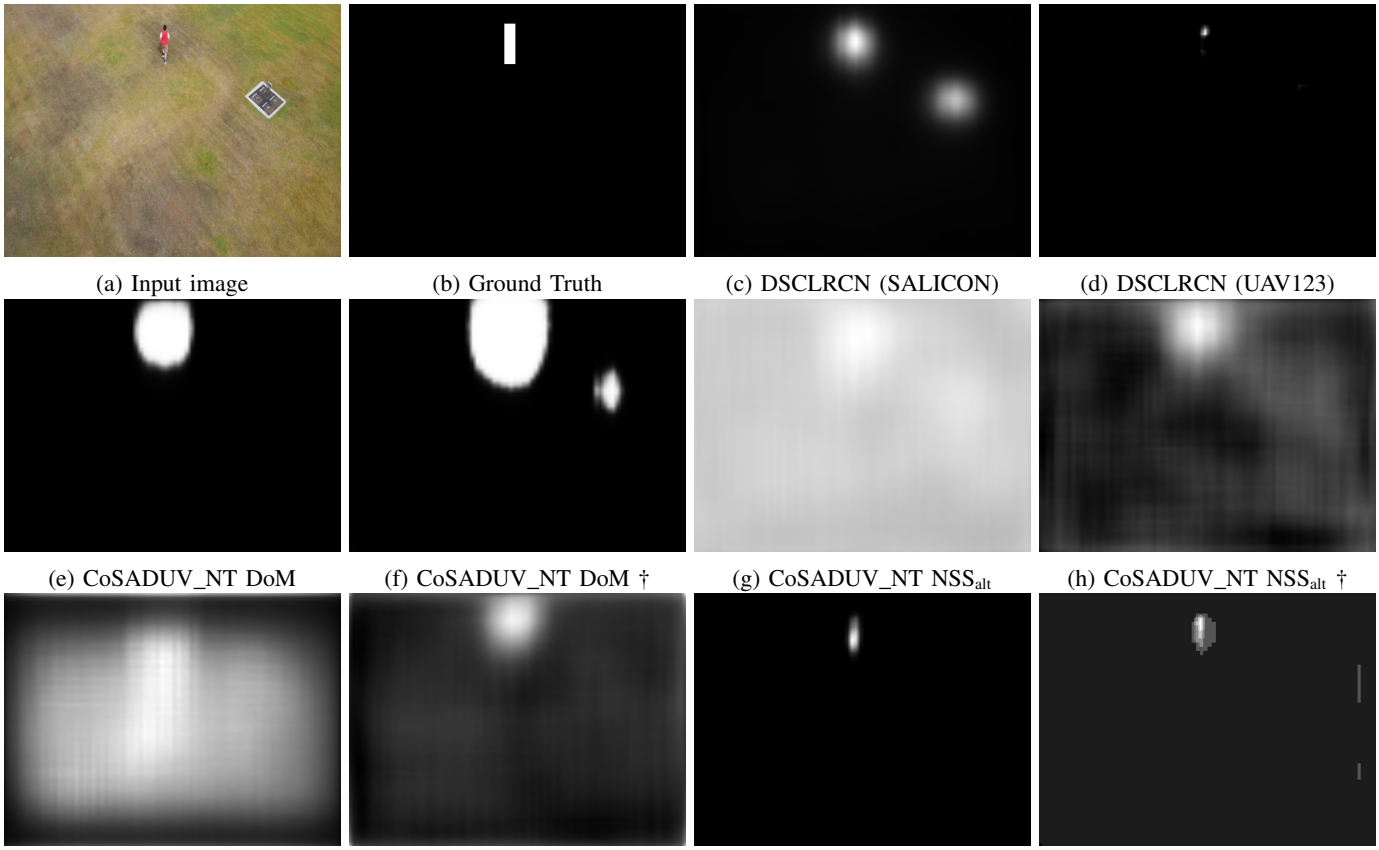
scattered throughout the prediction map. The CoSADUV_NT model trained using the DoM loss function similarly detects both objects with high confidence, also labelling a large part of the surrounding area as salient. The other CoSADUV_NT models struggle to accurately detect the salient objects, either detecting nothing or producing high numbers of false positives.

Model Architecture	Training Loss Function	↑NSS_{alt}
DSCLRCN	NSS	2.960
DSCLRCN	NSS_{alt}	3.074
CoSADUV_NT	DoM	3.123
CoSADUV_NT †	DoM	1.079
CoSADUV_NT	NSS_{alt}	1.190
CoSADUV_NT †	NSS_{alt}	−0.122
CoSADUV_NT	CE_MAE	2.032
CoSADUV 1×1	NSS (1 frame)	0.701
CoSADUV 3×3	NSS (1 frame)	20.789
CoSADUV 3×3	NSS (2 frames)	24.065

TABLE II: Quantitative scores of the tested model architecture variations on the video frame shown in Figure 8.

Out of the temporal models, CoSADUV (3, 1) and CoSADUV (3, 2), Figure 8k and 8l respectively, achieve very similar results with the former model selecting a slightly larger area as salient than the latter. The CoSADUV (1, 1) model in Figure 8j has a small peak at the locations of both the salient objects, but the prediction map produced is very noisy. Out of all the models, the two temporal models with a 3×3 kernel produce the most accurate and well-defined saliency prediction maps. This is reflected in the NSS_{alt} scores for this image, which are provided in Table II. Notably, the model trained with NSS_{alt} and further trained on the EyeTrackUAV dataset achieves a negative NSS_{alt} score on this frame, indicating that it completely fails to detect the target object. Seemingly this is because this model identifies the boat as the most salient object in the image but does not identify the wakeboarder, which is the only object labelled in this image.

Another image with qualitative results is shown in Figure 9. This input image is from the 'person7' sequence, which we placed in our test set. The performance of each model on this image is similar to that in Figure 8, with some notable differences. Firstly, the DSCLRCN model trained on the UAV123 dataset performs significantly better on this image, only detecting the salient object of the person in a red coat in the top-centre of the image, without any false positives scattered around the image. This model still under-detects the salient object, however. The CoSADUV_NT model trained using the DoM loss function again detects only the salient object(s), with fairly significant over-detection around



(a) Input image (b) Ground Truth (c) DSCLRCN (SALICON) (d) DSCLRCN (UAV123)

(e) CoSADUV_NT DoM (f) CoSADUV_NT DoM † (g) CoSADUV_NT NSS_{alt} (h) CoSADUV_NT NSS_{alt} †

(i) CoSADUV_NT CE_MAE (j) CoSADUV 1×1 (1 frame) (k) CoSADUV 3×3 (1 frame) (l) CoSADUV 3×3 (2 frame)

Fig. 9: Qualitative results of the different models tested on an image our UAV123 test set. The input image is frame 96 in the 'person7' sequence in UAV123 [29].

the object. The same model trained with the CE_MAE loss function does show a small peak in the area of the image close to the salient object, however the prediction output is very noisy. The CoSADUV (1, 1) model, Figure 9j, performs significantly better on this image than in the one shown in Figure 8, however some false negatives are visible throughout the image. The CoSADUV (3, 1) model (Figure 9k) produces a similarly accurate prediction as in the last image, but the CoSADUV (3, 2) model is visibly worse, producing a saliency prediction map which is both noisy and blocky.

The performance of each of these models on this frame for the NSS_{alt} metric is shown in Table III. These show that again the two CoSADUV models with temporal implementation using a 3×3 kernel in the convLSTM layer significantly outperform all other models. However, in this image the CoSADUV model trained with 2-frame backpropagation produces somewhat noisy output, shown in Figure 9l. The predicted salient area at the location of the person is wider than expected, and wider than the prediction in Figure 9k, and there are some noise salient predictions present.

Figure 10 presents a comparison of the best non-temporal model (CoSADUV_NT trained with DoM loss function) and the best temporal model (CoSADUV (3, 1)) throughout a video sequence. As expected, the temporal model performs

Model Architecture	Training Loss Function	↑NSS_{alt}
DSCLRCN	NSS	6.755
DSCLRCN	NSS_{alt}	4.160
CoSADUV_NT	DoM	5.417
CoSADUV_NT †	DoM	3.597
CoSADUV_NT	NSS_{alt}	3.814
CoSADUV_NT †	NSS_{alt}	4.554
CoSADUV_NT	CE_MAE	0.556
CoSADUV 1×1	NSS (1 frame)	4.977
CoSADUV 3×3	NSS (1 frame)	11.376
CoSADUV 3×3	NSS (2 frames)	10.529

TABLE III: Quantitative scores of the tested model architecture variations on the video frame shown in Figure 9.

significantly better in terms of accuracy of saliency prediction, detecting the salient object in all but one of the frames. The CoSADUV_NT model fails to detect any salient object in frame 90 and produces poor predictions for frame 0 and 30. The temporal model outperforms the non-temporal one in the consistency of saliency prediction in consecutive frames. In the 120 frame span shown in this figure the car moves slightly from the right to the left, but does not change in scale. The saliency prediction of the temporal model reflects this as the main predicted area remains constant in size. The non-temporal model, however, does not show the same level of consistency. Throughout the sequence shown this model

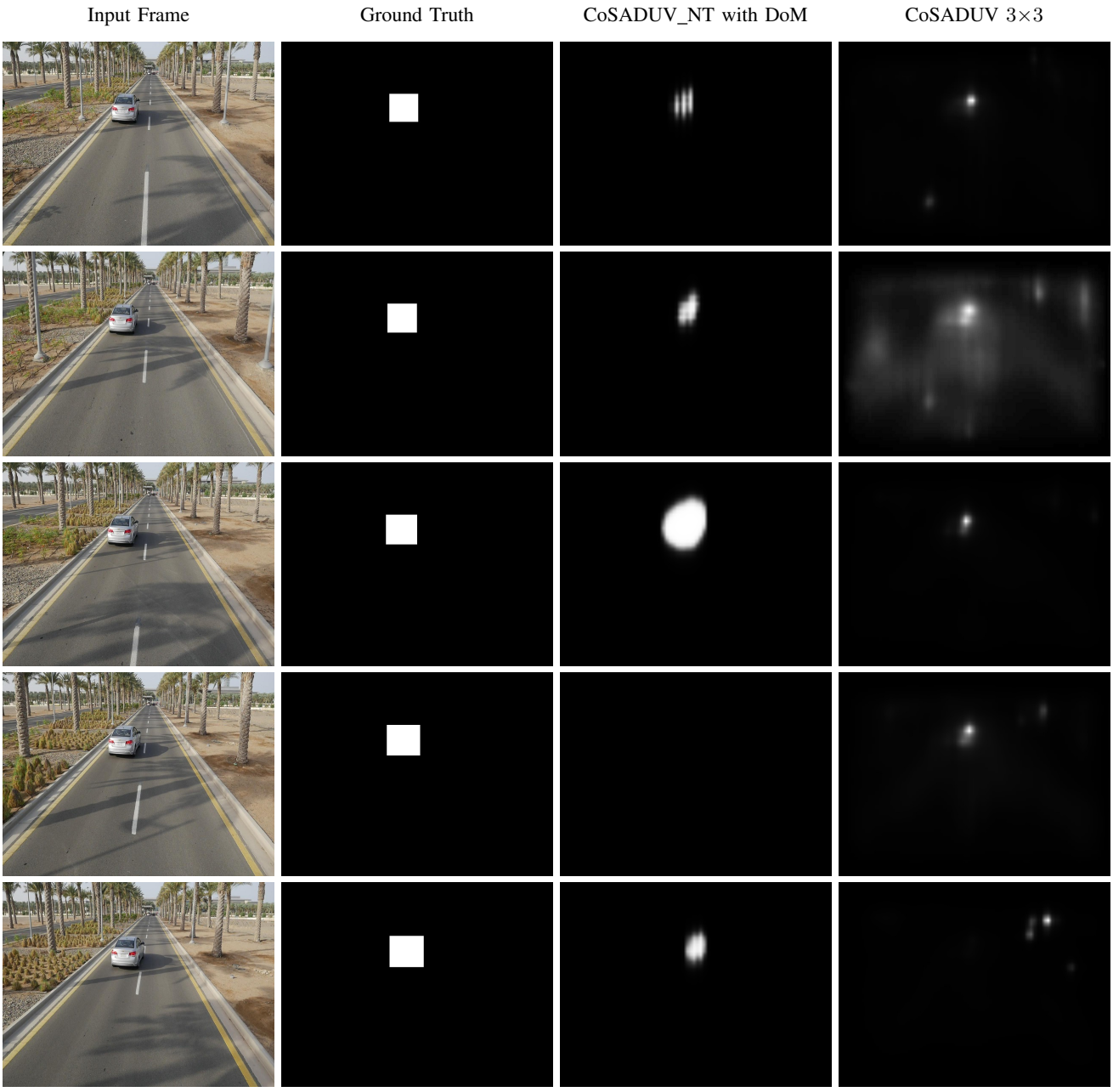
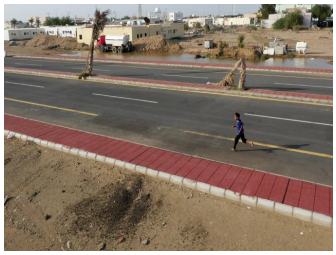


Fig. 10: Comparison of performance of temporal and non-temporal models over the duration of a sequence. Depicted are the first frame as well as every 30th subsequent frame, with corresponding ground truth and predictions by the CoSADUV_NT model trained with DoM loss function as well as the CoSADUV temporal model with 3×3 kernel size and trained with 1 frame backpropagation. Input images are from the 'car6' sequence in the UAV123 dataset [29].

outputs varying sizes of predicted areas, with the first output having significant gaps in saliency at the location of the object. Only in frame 120 does the non-temporal model outperform the temporal model, as the latter erroneously detects a salient object in the top-right of the image.

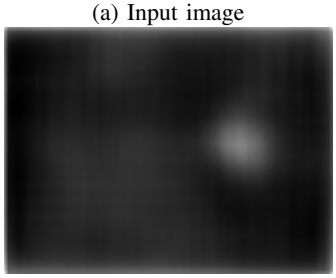
In Figure 11 we qualitatively compare the performance of the temporal CoSADUV models when applied to a single frame and when applied to a video sequence. (Note: the predictions presented above in Figure 8 and 9 are the video-based predictions.) The input sequence used is the 'person23' sequence, with results shown for frame 120.

As expected all three temporal models show an improvement in the predicted saliency map when applied to the entire video sequence leading up to this frame. Processing the previous frames allows the model to include the context of the previous frames through our temporal implementation when evaluating the saliency of the frame. Surprisingly, however, the CoSADUV (3, 1) model and the CoSADUV (3, 2) model perform very similarly when applied to only the given frame, however adding the video information slightly improves

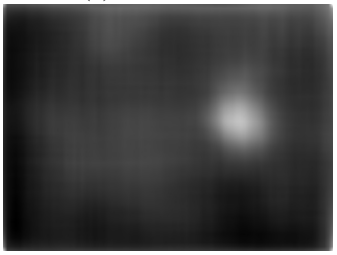


(a) Input image

(b) Ground Truth

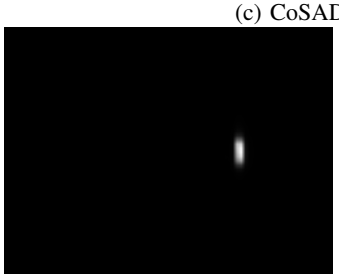


Frame-based prediction

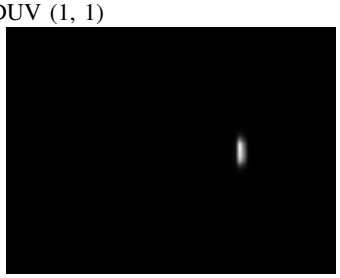


Video-based prediction

(c) CoSADUV (1, 1)

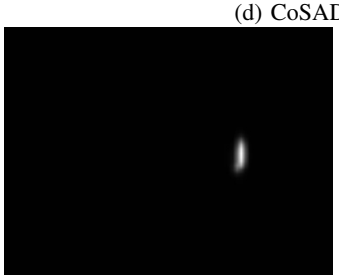


Frame-based prediction

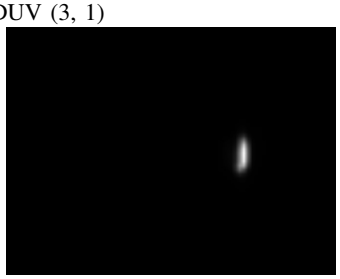


Video-based prediction

(d) CoSADUV (3, 1)



Frame-based prediction



Video-based prediction

(e) CoSADUV (3, 2)

Fig. 11: Impact of temporal information on the performance of CoSADUV models. Predicted saliency maps are scaled to $[0, 1]$. The video used is the 'person23' sequence from [29].

the prediction as both models become more confident and favour the left-side of the person over their back. Overall our proposed CoSADUV (3, 1) model achieves the best results out of all models tested, significantly outperforming both the DSCLRCN and CoSADUV_NT models.

V. Evaluation

In Table I we present the performance of the models we tested on a range of performance metrics. Although the CoSADUV_NT model trained with CE_MAE loss function achieves the best score in three out of the five metrics reported, the qualitative results show that this model does not perform well in practice. The reason for this lies with the loss function used, as this is the only difference between this model and the CoSADUV_NT model trained with DoM loss function which achieves promising results. The issue likely lies with the fact that the target salient objects in our training data are in almost all cases very small in the image, and thus by simply outputting a very low predicted value at all spatial locations the model can achieve numerically good results for metrics such as CE and MAE, and hence also for the loss function CE_MAE. The model therefore fails to learn any meaningful patterns in the data. This result shows that while the CE and MAE metrics may have a place in evaluating the performance of models, and have been used successfully to train video saliency models such as [18], they are not suitable to be used as loss functions for training a model for our task.

Similarly, using the NSS loss function to train a saliency prediction model is strongly supported and suggested in literature, for example by [31], amongst other reasons because it is equally sensitive to false positives and false negatives. As previously discussed we altered the NSS function to be able to apply it to the dataset used, resulting in the NSS_{alt} function. The NSS_{alt} loss function, however, also has several limitations. Firstly, due to it using a piecewise function to evaluate images depending on the ground truth, and the range of the two functions it is composed of being different, its value significantly changes depending on how the test set is composed. If the test set contains a large number of negative examples, the expected value for the NSS_{alt} function for a 'good' model approaches 0, as the lowest score possible for a negative image is 0. For this reason we split the two components of the NSS_{alt} function up when reporting the scores of the tested models in Table I. Additionally, many of the models tested were trained with the NSS_{alt} loss function. However, in the case of CoSADUV_NT, training the model with the DoM loss function yielded better performance, even in the $\text{NSS}_{\text{alt}}(+)$ performance metric.

Identifying the above issues with using the CE_MAE and the NSS_{alt} loss functions for training our model inspired us to create and investigate the DoM loss function. This loss function produced the best result for the CoSADUV_NT model, and is a useful metric for analysing the performance of the models as it compares the average saliency prediction for target and non-target locations. However, the loss function has a major issue for use for training models. Namely, as the loss function has a range of $[-1, 1]$, models risk reaching a training loss value $l \approx 0$, where the gradient produced by the loss function is extremely small. This may make it difficult for the model to update its weights, making it unable to improve continue learning. In the case of the CoSADUV_NT model, this was overcome by setting a higher initial learning rate. However, we were unable to achieve the same for the CoSADUV model, hence no results are presented for this model and loss function. To summarize, due to the characteristics of UAV anomaly detection data loss functions commonly used for training saliency and video saliency prediction models

may not be applicable or produce good results, which may limit the potential of deep learning approaches to this problem. Despite this, our model was able to achieve better results than the state-of-the-art in non-UAV saliency prediction.

Another factor limiting the potential of deep learning approaches to the problem of anomaly detection in UAV video is the lack of public datasets. As discussed in the Solution section, no public dataset was found which provided labelled data for anomaly detection in UAV video. A single dataset, EyeTrackUAV, was found which provides saliency labels for a set of UAV video sequences; however, these labels are produced from human eye fixation data, which is distinct salient object identification labels. The data in EyeTrackUAV was judged to be less accurate for the task of salient object identification than the object tracking data provided with UAV123 and so this was used instead, despite these labels also having shortcomings for our task. As the data is labelled for object tracking, only a single object is labelled throughout each sequence. For some of the sequences this is not an issue, as the only salient object present in the input sequence is the tracked object. Many of the sequences, however, contain several salient objects moving in and out of the video which are not included in the labels. This means that the labels provided with the UAV123 dataset are inconsistent for the task of salient object detection which will compromise the validity of performance metrics collected for this dataset, and will produce less generalizable models.

For this reason even though the proposed model achieves impressive results on the UAV123 dataset, it may not achieve as good results when applied to other data. An example of this is shown in Figure 12, which shows two cropped images from an image in the HERIDAL dataset produced by [14]. The first image contains no salient objects and the second image contains three people fully in view, with a fourth on the edge in the bottom-right. These images show that both models struggle with the data as it is generally more cluttered and contains less uniform backgrounds than the UAV123 dataset. The non-temporal model successfully predicts no salient object present in the first image, but then misses all salient objects in the second image, instead predicting an area in the bottom-left as salient. The CoSADUV model incorrectly predicts the presence of a salient area in the top-left of the first image, and produces very noisy predictions in the second image. However, this model arguably detects the two people near the centre of the image, although very poorly, as these contain small peaks in the saliency prediction. Also, as previously noted the HERIDAL dataset only contains images, thus the CoSADUV model is not able to operate optimally on this data.

The results do, however, show that our approach has merit, and given a more accurate dataset the model is likely to achieve even better results. This dataset could simply be produced by augmenting the existing labels for the UAV123 dataset by adding labels for the other salient objects in each sequence. However, this would be a very time-consuming task, and we were unable to carry it out in the scope of this project.

Another limitation of our proposed solution is its processing speed. As previously stated, when ran on a GeForce RTX 2080 Ti GPU the CoSADUV model ran at 2.2 FPS. This is too slow to achieve real time processing: the UAV123 dataset was collected at 30 FPS, and additionally provided downsampled data at 10 FPS. It is however arguably fast enough to be useful in a practical setting. This execution speed was achieved without any parallel processing of videos or frames, and only used 3200MB of the 10989MB of memory available on the GPU. It is therefore possible that the model could be optimised with parallelisation in order to achieve faster processing. This limitation could also be overcome by using a different base model, applying the same or similar modifications as we did to the DSCLRCN model in this project. Although the DSCLRCN model achieves good results, it is slow compared to some other state-of-the-art models such as the Non-Local Deep Features model by [13] due to its use of LSTM layers to scan the image.

Lastly, the proposed model is only designed to process images at a resolution of 640×480. This still achieves good results on the UAV123 dataset, which is natively captured at 1280×720, however the same may not be true for larger images. For example, the HERIDAL dataset features images of size 4000×3000 which contain very small salient objects in the form of people. In order to process such images, our proposed model would either need to downscale the image, potentially removing some salient objects from it due to the extremely small scales of objects present, or crop the image and process one section of the original image at a time, which would be extremely inefficient. As the model only requires 3200MB of the $>$ 10000MB GPU memory available on the hardware used for training it is possible that the input size of the model could be increased to accommodate larger images. This was not investigated in this project, as the spare memory available on the GPU was instead used to speed up training by increasing the number of images processed on the GPU at once, or for deeper recursion into previous frames during back propagation of the temporal model.

The factors above likely limit the potential of our proposed solution to be applied for the task of anomaly detection in UAV video. Although the model achieves very good results on the UAV123 dataset, it does not generalize well to the data presented in the HERIDAL dataset. Despite this, the model produced and our approach to this task have proven merit, and given more time to create a dataset with more proper target data it is possible a generalizable model could be produced.

VI. Conclusions

UAV can be great tools for use in search and rescue operations but their potential is limited by a requirement of slow and expensive human analysis of the recorded video data. In order to solve this problem we propose a deep learning approach based on the DSCLRCN model by [5], a state-of-the-art model for salient object detection using a deep convolutional neural network (CNN) approach. Our proposed solution is novel in applying a deep learning approach to visual saliency for the purpose of anomaly detection in UAV video, and in considering both local context in the image as well as

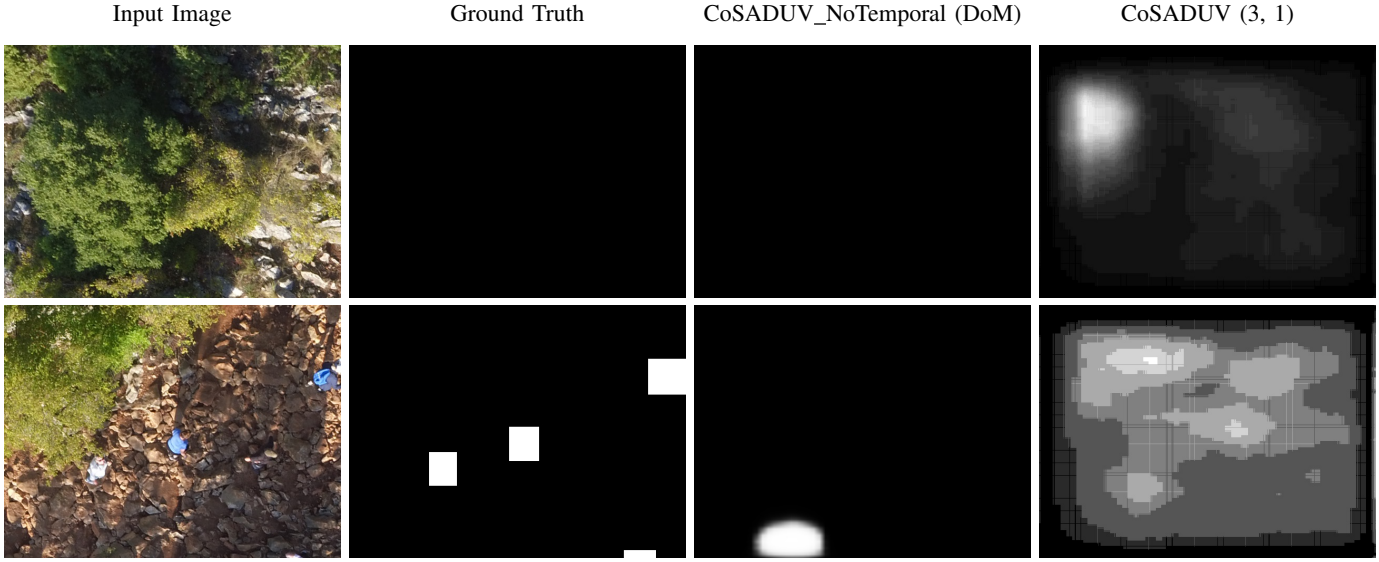


Fig. 12: Analysing the performance of the best CoSADUV_NoTemporal model and the best CoSADUV model on cropped images from the HERIDAL dataset [14]. The top image contains no salient objects while the bottom contains three people fully in the image, and a fourth on the edge. The performance of the proposed method of [14] on the full image can be found in Figure 3 in [14].

the general scene context of the image as a whole. It is also novel in leveraging the temporal information carried in video captured by UAV, which previous approaches have not done. In this way we present a novel approach to the problem of anomaly detection in UAV video by using contextual saliency and temporal information.

In order to achieve better performance we modify the DSCLRCN model by using a *Sigmoid()* activation function, and train the model with our novel NSS_{alt} and DoM loss functions. We add temporal processing to our model by adding a convolutional LSTM layer to propagate saliency features extracted at each spatial location in the image through time allowing the model to learn the temporal consistency of data in video, producing our proposed Contextual Saliency for Anomaly Detection in UAV Video (CoSADUV) model.

We compare the performance of the CoSADUV model with the baseline DSCLRCN model, as well as a version of the model without temporal processing (CoSADUV_NoTemporal). For each model several variations are compared in order to investigate different loss functions, datasets, and network settings. As no public dataset exists for anomaly detection in UAV video we use the UAV123 dataset for object tracking in UAV video [29], which we split into training, validation and testing sets. The object tracking labels approximate the task of anomaly detection in many of the videos. The baseline DSCLRCN model trained on the SALICON dataset for visual saliency detection [34] achieves an NSS_{alt} score of 3.552 on positive images (containing at least one target salient pixel), and an NSS_{alt} score of 0.091 on negative images. The best CoSADUV_NT model is produced by using the DoM loss function. This achieves an NSS_{alt} of 3.315 on positive images and 0.163 on negative images. Despite these quantitative scores being lower than the DSCLRCN model, the CoSADUV_NT model produces better qualitative results in many cases. The best CoSADUV model uses a kernel size of 3×3 in the convolutional LSTM layer and is trained by limiting backpropagation of the convolutional LSTM layer to one frame backwards in time. This model achieves an NSS_{alt} of 8.851 on positive images and 0.023 on negative images, significantly outperforming the other two model architectures.

While the results achieved by the proposed CoSADUV model are very promising, the model generalizes poorly to other datasets such as HERIDAL [14]. This is caused in part by shortcomings in the dataset used to train the model. Some of the sequences provided in the UAV123 dataset contain only a single salient object throughout the sequence, meaning the labels provided with the dataset are accurate for the task of salient object detection. However, most of the sequences contain other salient objects which are not labelled in the ground truths, producing data which is inconsistent with our goal. This means that while the model achieves good test results on our UAV123 test set, the same may not be true for other data. Our model fails to beat the model presented by [14], which is the state-of-the-art in people detection in UAV images for search and rescue. However, their solution targets the narrow scope of people detection in land search and rescue operations rather than the wider scope of of this project.

If given extra time the results achieved by our proposed model could likely be improved, and its generalizability drastically improved, by producing more accurate and consistent labels for the UAV123 dataset. Currently the lack of a publicly available, labelled dataset for UAV anomaly detection severely reduces the effectiveness of any deep learning based approach

to this task. As such approaches become more common in more and more fields within computer vision, this poses a significant limiting factor in this field.

Future research could build on our work in a number of directions. Firstly, the portability of our approach to this task could be investigated by attempting to apply the same approach to a different model for visual saliency detection. If successful this could yield improvements in terms of processing speed and memory usage as well as saliency prediction accuracy by leveraging other state-of-the-art methods available. Secondly, different temporal processing implementations could be investigated. For example, alternative architectures combining recurrent and convolutional layers could be used either in the last layer as in this work or in another stage of the model. An extensive investigation could be carried out into which stage of the model most benefits from temporal information processing, and whether constructing a deeper temporal model yields further improvements in accuracy.

REFERENCES

[1] S. Friess, "50,000 Volunteers Join Distributed Search for Steve Fossett — WIRED," 2007. [Online]. Available: https://www.wired.com/2007/09/50000-volunteers-join-distributed-search-for-steve-fossett/

[2] The Associated Press, "Nevada wants Fossett widow to pay search cost - US news - Life — NBC News," 2008. [Online]. Available: http://www.nbcnews.com/id/24419374/ns/us_news-life/t/nevada-wants-fossett-widow-pay-search-cost/

[3] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, nov 1998.

[4] J. Sokalski, T. P. Breckon, and I. Cowling, "Automatic Salient Object Detection in UAV Imagery," *Proc. 25th International Conference on Unmanned Air Vehicle Systems*, pp. 11.1–11.12, 2010.

[5] N. Liu and J. Han, "A Deep Spatial Contextual Long-term Recurrent Convolutional Network for Saliency Detection," *CoRR*, vol. abs/1610.0, 2016.

[6] Y. Zhang, A. Su, Z. Xianwei, X. Zhang, and Y. Shang, "Salient Object Detection Approach in UAV Video," *Proc SPIE*, vol. 8918, 2013.

[7] S. Gotovac, V. Papić, and Ž. Marušić, "Analysis of saliency object detection algorithms for search and rescue operations," in *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, sep 2016, pp. 1–6.

[8] C. Wang and B. Yang, "Saliency-guided object proposal for refined salient region detection," *2016 Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2016.

[9] Y. Zhang, X. Wang, X. Xie, and Y. Li, "Salient Object Detection via Recursive Sparse Representation," *Remote Sensing*, vol. 10, no. 4, p. 652, 2018.

[10] L. Wang, J. Xue, N. Zheng, and G. Hua, "Automatic salient object extraction with contextual cue," in *2011 International Conference on Computer Vision*, nov 2011, pp. 105–112.

[11] X. Yang, X. Qian, and Y. Xue, "Scalable Mobile Image Retrieval by Exploring Contextual Saliency," *IEEE Transactions on Image Processing*, vol. 24, pp. 1709–1721, 2015.

[12] I. Rahman, C. Hollitt, and M. Zhang, "Contextual-based top-down saliency feature weighting for target detection," *Machine Vision and Applications*, vol. 27, no. 6, pp. 893–914, aug 2016.

[13] Z. Luo, A. K. Mishra, A. Achkar, J. A. Eichel, S. Li, and P.-M. Jodoin, "Non-local Deep Features for Salient Object Detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6593–6601, 2017.

[14] D. Božić-Štulić, Ž. Marušić, and S. Gotovac, "Deep Learning Approach in Aerial Imagery for Supporting Land Search and Rescue Missions," *International Journal of Computer Vision*, pp. 1–23, mar 2019. [Online]. Available: http://link.springer.com/10.1007/s11263-019-01177-1

[15] N. Imamoglu, W. Lin, and Y. Fang, "A Saliency Detection Model Using Low-Level Features Based on Wavelet Transform," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 96–105, jan 2013.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Tech. Rep., 2015. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/results

[17] A. Azaza and A. Douik, "Deep saliency features for video saliency prediction," in *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, mar 2018, pp. 355–359.

[18] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam, "Pyramid Dilated Deeper ConvLSTM for Video Salient Object Detection," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 744–760.

[19] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *NIPS*, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Tech. Rep., 2015. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," sep 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," Tech. Rep., 2009. [Online]. Available: http://www.image-net.org.

[23] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning Deep Features for Scene Recognition using Places Database," Tech. Rep., 2014. [Online]. Available: http://places.csail.mit.edu

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, 2012.

[25] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "IEEE Transactions on Pattern Analysis and Machine Intelligence Places: A 10 million Image Database for Scene Recognition," 2017. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/index.html

[26] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015, pp. 3431–3440. [Online]. Available: http://ieeexplore.ieee.org/document/7298965/

[29] M. Mueller, N. Smith, and B. Ghanem, "A Benchmark and Simulator for UAV Tracking," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2016.

[30] R. J. Peters, A. Iyer, L. Itti, and C. Koch, "Components of bottom-up gaze allocation in natural images," *Vision Research*, vol. 45, no. 18, pp. 2397–2416, aug 2005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0042698905001975

[31] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand, "What Do Different Evaluation Metrics Tell Us about Saliency Models?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 3, pp. 740–757, mar 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8315047/

[32] J. Li, C. Xia, Y. Song, S. Fang, and X. Chen, "A data-driven metric for comprehensive evaluation of saliency models," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter. IEEE, dec 2015, pp. 190–198. [Online]. Available: http://ieeexplore.ieee.org/document/7410387/

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2015.

[34] M. Jiang, S. Huang, J. Duan, and Q. Zhao, "SALICON: Saliency in Context," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2015.

[35] V. Krassanakis, M. Perreira Da Silva, and V. Ricordel, "Monitoring Human Visual Behavior during the Observation of Unmanned Aerial Vehicles (UAVs) Videos," *Drones*, vol. 2, no. 4, 2018. [Online]. Available: http://www.mdpi.com/2504-446X/2/4/36